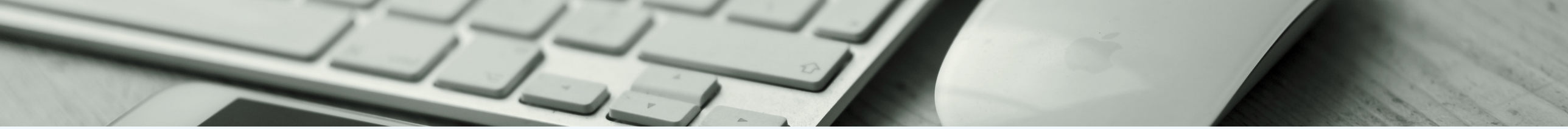
ASSO - HOMEWORK 06

Presented by Group 33

SUMMARY OF CONTENTS

OUR MAIN TOPICS TODAY

Background
Architecture
Other Architectural Patterns
Quality Attributes
Conclusion
Q&A



BACKGROUND

WHY WAS NGINX CREATED?

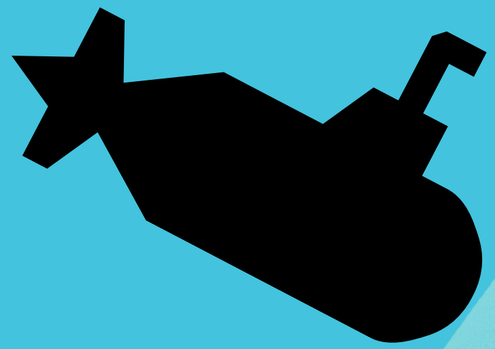
- Web server technology closely followed the architecture of the Internet and organizational network demands.
- The web-server landscape was dominated by Apache, which relied on a forking model. This started getting inefficient as hardware improved.
- The **need for better technologies** increased.

...this led to the creation of Nginx

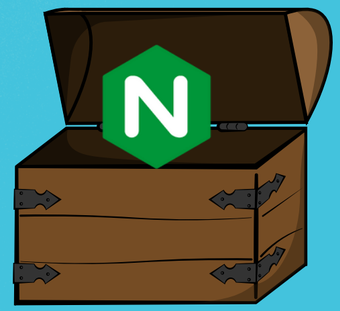
~42%

Nginx’s market share (in web-and-application-servers market)

IS THIS JUST LUCK?



LET'S DIVE **DEEPER...**



ARCHITECTURE

- Nginx is built around an **Event-Driven** architecture.
- There is one master process and several worker processes, which are forked at start-up to reduce runtime allocations and context switches.
- Communication between processes happens using message-passing built on top of shared-memory mechanisms.
- Configuration read by the master process and made available in a shared-repository fashion.
- Modular design which enables having a small core which can be extended by multiple plug-ins;

ARCHITECTURAL OVERVIEW

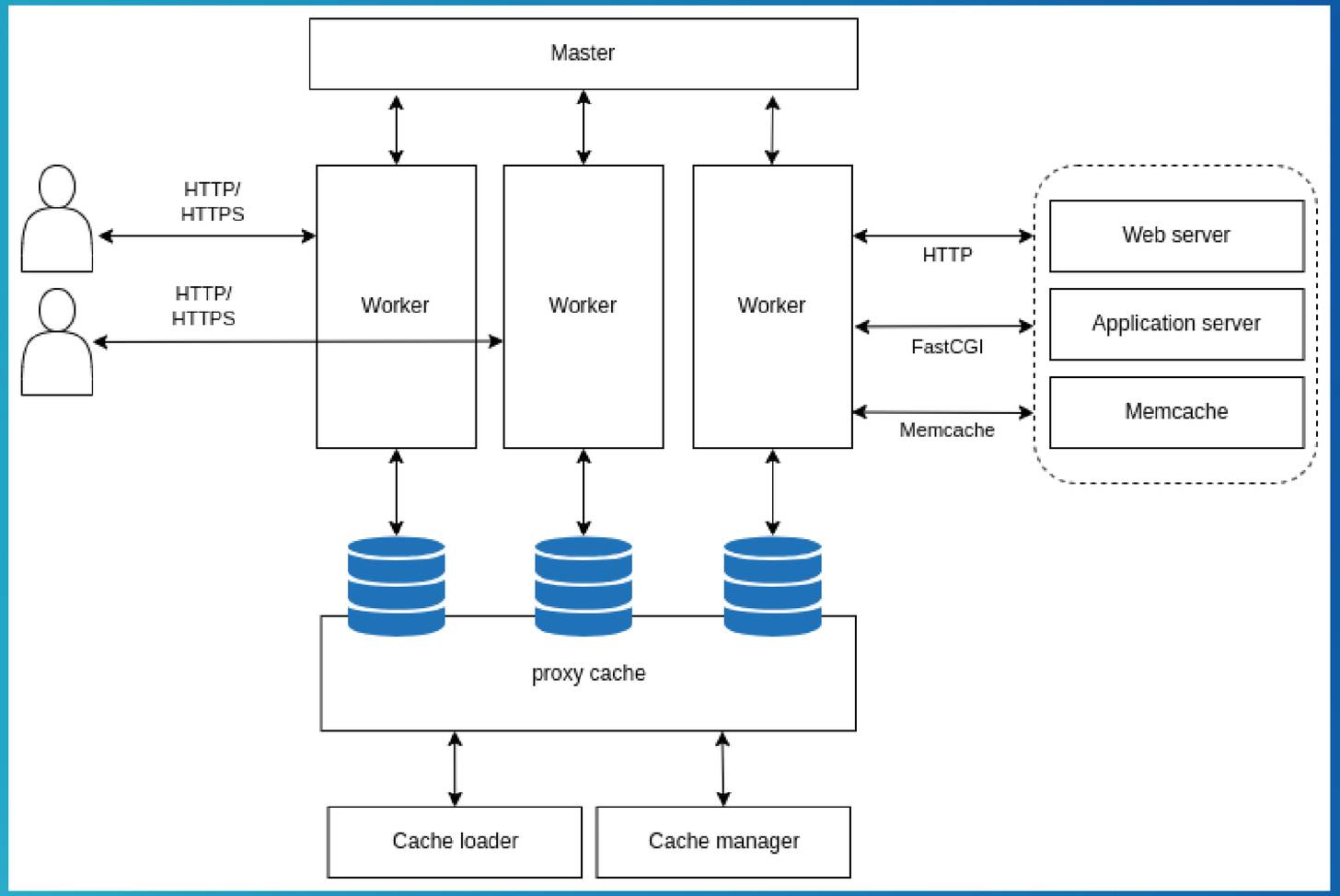


Figure 1: Nginx's architectural overview

Architectural Patterns

PLUGIN	SHARED REPOSITORY	PIPES AND FILTERS	INTERCEPTOR	INTERPRETER
Application is structured in a way that allows pieces of its functionality, termed as 'plugins', to be added and removed seamlessly	Indirect form of communication where components do not know each other	Each filter performs a specific task on the data that flows through the pipeline, and the output of one filter becomes the input of the next filter (modularity, flexibility, reusability)	Used when software systems or frameworks want to offer a way to change, or augment, their usual processing cycle	Interprets a higher level language to a lower level language to directly execute a series of commands, without requiring programs being compiled ahead of time

Table 1: nginx’s architectural patterns and basic definitions

BUT...

WHERE ARE THIS PATTERNS PRESENT IN NGINX?

PLUGIN PATTERN

Nginx ’s modular architecture lets developers extend the set of features without changing the core.

The modules act as plugins that can be added at build time to enhance the functionality of the web server.

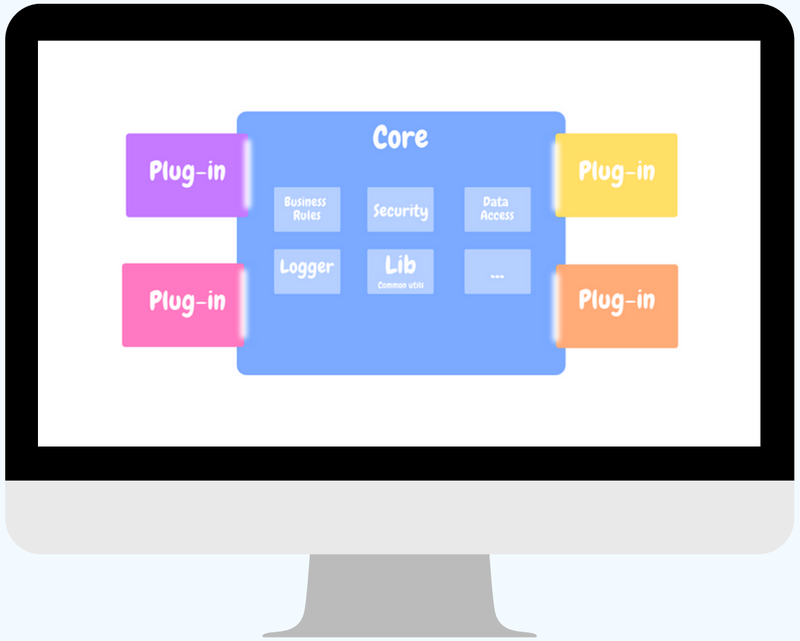


Figure 2: Plugin pattern

SHARED REPOSITORY PATTERN

After reading the configuration file (or files, as Nginx allows configuration to be split across files for better modularity), the master process allows worker processes to get a read-only view of the loaded configuration, effectively serving as the central source of truth for an instance’s configuration.

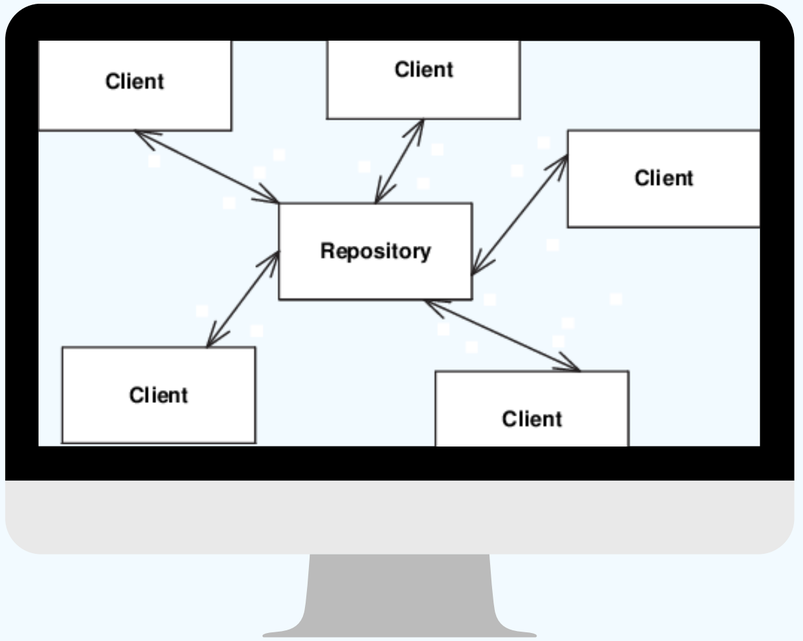


Figure 3: Shared Repository pattern

PIPES AND FILTERS

Nginx's worker code has two main parts: core and functional modules.
The core controls data flow and executes module code during request processing.

Functional modules act as pipeline filters, each performing specific tasks and work in sequence, with each module's output becoming the input for the next step.

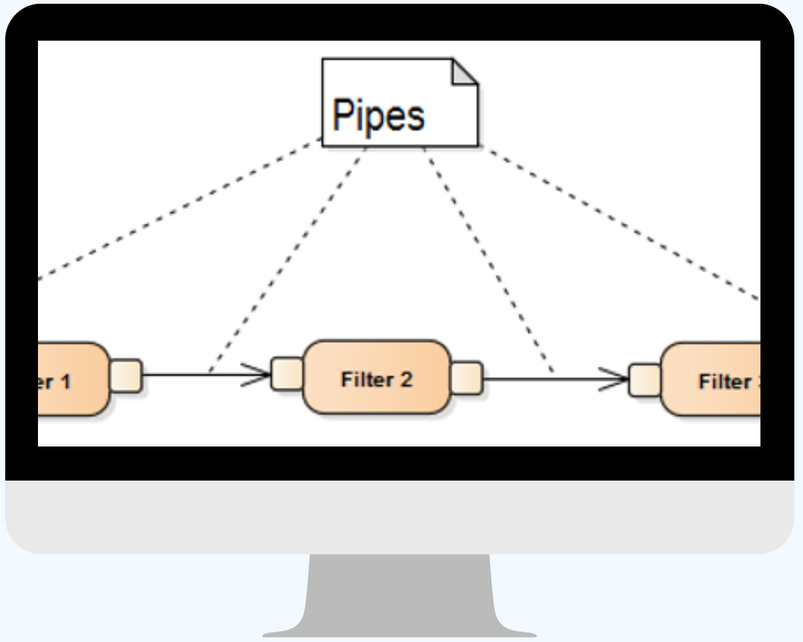


Figure 4: Pipes and Filters pattern

INTERCEPTOR

Separate processes handle different stages of connection processing within a highly efficient run-loop.

Workers are designated specific tasks, allowing Nginx to intercept incoming connections, process them and manage concurrent requests.

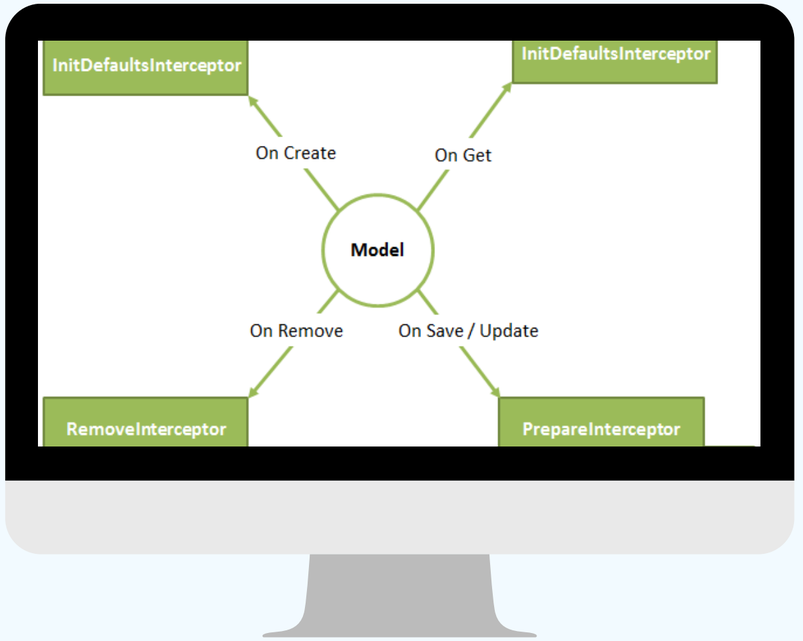


Figure 5: Interceptor pattern

INTERPRETER

The standard Nginx distribution supports embedding Perl scripts only.

Perl is a high-level, general-purpose, interpreted, dynamic programming language.

The Interpreter specifies how to evaluate Perl sentences.

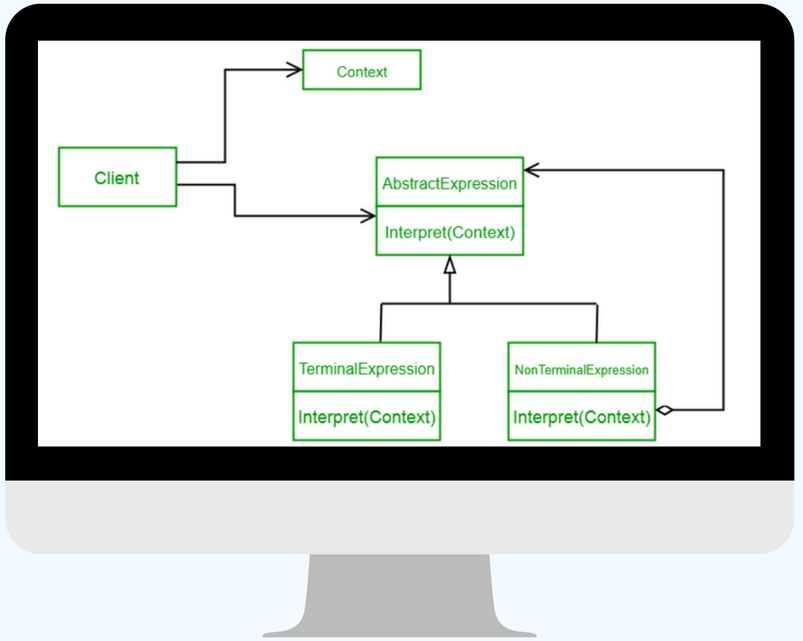


Figure 6: Interpreter pattern

QUALITY ATTRIBUTES [1/2]

- **Configurability** – Nginx is configured through config files, including added modules.
- **Scalability** - deliver tens of thousands of concurrent connections on a server.
- **Modularity** - modules allow the extension of Nginx 's core. There are different types of modules: core modules, event modules, phase handlers, protocols, variable handlers, filters, upstreams and load balancers.
- **Extensibility** - new features were added, use of distributed memory object caching systems and reverse proxies. Modules can be developed to extend Nginx 's core functionalities.

QUALITY ATTRIBUTES [2/2]

- **Modifiability** - the code is open source, so anyone can modify it to adapt to their own use.
- **Usability** - config files follow a C-style syntax and formatting, making it easy to use for the majority of Nginx's end users.
- **Maintainability** - the config files can be easily automated, due to following C-style conventions.

CONCLUSION

- **Innovative event-driven architecture.**
- **Modular plugin model for extensibility.**
- **Scalable with shared repository pattern.**
- **Flexible pipes & filters implementation.**
- **Key attributes: configurability, scalability, modularity.**
- **Deepened understanding of architecture's success.**

QUESTIONS? COMMENTS?

LET US KNOW!

“

ALWAYS REMEMBER

Almost everything we know about good software architecture has to do with making software easy to change

---

MARY POPPENDIECK